можна виокремити декілька етапів: перший рівень – нормативно-документаційний; другий рівень – рівень інститутів; третій рівень – індивідуальний (особовий). На першому рівні органи державної влади повинні створити нормативно-документаційну базу, що враховує усі аспекти проблеми інформаційної безпеки. Другий рівень включає погоджену діяльність різних соціальних інститутів, пов'язаних з вихованням і соціалізацією, по забезпеченню інформаційної безпеки особи. Третій рівень пов'язаний, передусім, з самовихованням, самоосвітою, формуванням високого рівня інформаційної культури особи як частини загальної культури людини. На цьому рівні відбувається формування необхідних особових якостей для забезпечення інформаційного самозахисту особи.

Генча М.Е.
студент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ВЕБ-СОКЕТИ – НОВИЙ РІВЕНЬ СПІЛКУВАННЯ БРАУЗЕРА І СЕРВЕРА

Вебсокети(WebSockets) це просунута і відносно нова технологія, що дозволяє відкрити постійне двонаправлене мережеве з'єднання між браузером користувача та сервером. За допомогою його API ви можете відправити повідомлення на сервер і отримати відповідь без виконання HTTP запиту, причому цей процес буде подієво-керованим.

В чому ж унікальність вебсокетів ? Для того, що відповісти на це питання розглянемо ситуацію:

Припустимо, що Ви вирішили написати додаток для розсилки. По-перше, треба зробити клієнт, який буде перевіряти, чи є нові повідомлення кожну хвилину. Проте, більшу частину часу не було ніяких нових повідомлень, а клієнт щохвилини посилатиме нові запити, викликаючи величезне навантаження на сервер. Цей метод був дуже популярний, і називався Polling. Ним і зараз користуються, навіть великі компанії (наприклад Вконтакті).

Якщо логічно подумати, то єдиним правильним рішенням буде зробити так, що сервер відсилав повідомлення на клієнт наскільки швидко, наскільки це можливо. Тобто клієнт не має ініціалізувати запит, цим має займатися сервер. Це було неможливо впродовж довгого часу, але як тільки була представлена технологія вебсокетів, це, нарешті, стало можливим.

Коли ви починаєте думати, єдиний висновок, що ви можете зробити те, що сервер повинен послати повідомлення клієнта, як тільки є пошта доступна. Клієнт не повинен ініціювати запит, а сервер повинен зробити це. Це було

неможливо протягом тривалого часу, але так як WebSockets, де введені, це, нарешті, стало можливим.

WebSockets є протоколом і API JavaScript, протокол є дуже низькорівневим, повністю двостороннім протоколом, який означає, що повідомлення можуть бути відправлені в обох напрямках одночасно. Це дало можливість серверу відсилати дані на клієнт, замість того, щоб робити протилежне. Polling і Long Polling не були більше не потрібні.

Оскільки WebSockets забезпечують спосіб спілкування в обох напрямках, вони часто використовуються для додатків реального часу. Якщо, наприклад, хтось відкрив додаток і змінює деякі дані, ви можете безпосередньо оновити візуалізовані дані для всіх користувачів за допомогою WebSockets.

Як тільки сторінка сайту вирішила, що вона хоче відкрити WebSocket на сервер, вона створює спеціальний Javascript-об'єкт. Все починається так само як в звичайному HTTP-запиті. Браузер підключається по протоколу TCP на 80-й порт сервера, але дає трохи незвичайний GET-запит, і, якщо сервер підтримує WebSocket, то він відповідає. Якщо браузер це влаштовує, то він просто залишає TCP-з'єднання відкритим. Все – встановлення зв'язку скоєно, канал обміну даними готовий.

Як тільки одна сторона хоче передати іншій якась інформація, вона відправляє дата-фрейм наступного вигляду:

0x00, <рядок в кодуванні UTF-8>, 0xFF

Тобто просто рядок тексту – сукупність електронних даних, до якої спереду приставлений нульовий байт 0x00, а в кінці – 0xFF. Без заголовків і метаданих. Що саме відправляти, розробники повністю залишили на розсуд розробників сайтів: XML, JSON, простий текст.

Зараз розглянемо детальніше як працює протокол. За допомогою протоколу WebSockets так само можна передавати і бінарні дані, наприклад, картинки. Для них використовується інший дата-фрейм такого вигляду:

0x80, <довжина – один або кілька байт «,» тіло повідомлення>

«Один або кілька байт» – дуже хитрий спосіб вказівки довжини тіла повідомлення [1]. Щоб не створювати обмежень на довжину переданого повідомлення і в той же час не витрачати байти нераціонально, розробники протоколу використовували наступний алгоритм. Кожен байт у вказівці довжини розглядається по частинах: найстарший біт вказує, чи є цей байт останнім (0) або ж за ним є інші (1), а молодші 7 бітів містять власне дані.

Відповідно, як тільки виходить ознака бінарного дата фрейму 0x80, то береться наступний байт і відкладається його в окрему «скарбничку», потім на наступний байт – якщо у нього встановлений старший біт, то переноситься і його в «скарбничку»[1]. І так далі, поки вам не зустрінеться байт зі старшим бітом – 0. Це означить що цей байт – останній в покажчику довжини. Він також складається в «скарбничку». Тепер з усіх байт в «скарбничці» забирається старші біти і з'єднується в єдине ціле залишку. Ось це і буде довжина тіла повідомлення.

ККД такого протоколу наближається до 95% [2]. Різниця буде особливо помітною, якщо робити частий обмін невеликих блоків даних. Швидкість

обробки також наближається до швидкості чистого TCP-сокета – адже все вже готово – з'єднання відкрито – всього лише байти переслати.

У протоколу є свої плюси і мінуси. Причому плюсів істотно більше. Опишемо їх далі.

Швидкість і ефективність передачі забезпечує малий розмір переданих даних, який іноді навіть буде поміщатися в один TCP-пакет – тут, звичайно ж, все залежить від логіки розробника [3]. При цьому врахуйте, що з'єднання вже готове – не треба витрачати час і трафік на його встановлення.

Стандартність, яка усуне потребу в цілій низці.

Час життя канал – WebSockets не має обмеження на час життя в неактивному стані. Це означає, що більше не треба періодично оновлювати з'єднання, тому що його не має права закривати проксі-сервера. Значить, з'єднання може висіти в неактивному вигляді і не вимагати ресурсів. Правда, на сервері буде забиватися TCP-сокети. Для цього досить використовувати хороший мультиплексор, і нормальний сервер легко потягне до мільйона відкритих з’єднань.

Комплексні веб-додатки – в HTTP передбачено обмеження на число одночасних відкритих сесій до одного сервера[2]. Тому для безлічі різних асинхронних блоків на сторінці доводиться робити не тільки серверний, а й клієнтський мультиплексор. Це обмеження не поширюється на протокол WebSocket. Відкривається стільки, скільки необхідно.

Ще одна особливість: в якості єдиного дозволеного кодування обрана UTF-8. Тепер про мінуси.

В кінці листопада 2010 Адам Барт (Адам Барт) опублікував результати дослідження надійності WebSocket. За його результатами з'ясувалося, що в разі використання прозорих проксі-серверів можлива підміна кеша переданих даних. Тобто користувачі замість реальних даних може отримувати версію даних від зловмисника. Проблема виявилася досить серйозною. В результаті розробники Firefox і Opera оголосили, що до усунення проблем в майбутніх версіях їх браузерів підтримка WebSocket буде закрита, хоча вони і залишили можливість їх увімкнути.

В одному з проектів, що переводяться на цей протокол, з'ясувалося, що всі користувачі, хто використовував антивірус Avast в стандартній конфігурації, які не могли нормально працювати з додатком. У Avast за замовчуванням включений режим так званого «Веб-екрану», який вирішив, що WebSocket протокол поганий, і мовчки «різав» його.

Список використаних джерел:

1. RFC 6455 – The WebSocket protocol, стандарт протоколу.
2. HTML5 WebSocket API чорновик специфікації W3C.
3. WebSockets.org.